Case Study: Strangler Pattern at Blackboard Learn (2011)

Module 6.2 Assignment

Irene Carrillo Jaramillo

CSD 380

June 29, 2025

In 2011, Blackboard Learn, one of the most widely used learning management systems in higher education, faced a significant challenge. Their core platform had evolved into a large, monolithic Java application over the years, one that had become difficult to manage, costly to maintain, and risky to change. Faced with increasing pressure to modernize while still supporting a live product used by millions, Blackboard's development team turned to a solution known as the Strangler Fig Pattern.

The central issue with Blackboard Learn’s legacy system was its architecture. Over time, it had grown organically, with tight coupling between components, limited modularity, and very few automated tests. Developers were increasingly reluctant to make changes due to the high probability of introducing bugs in unrelated parts of the application. New feature releases were slow, and the company’s ability to respond to market needs was severely limited. Rather than opting for a “big bang” rewrite, Blackboard adopted the Strangler Fig Pattern, a method popularized by software architect Martin Fowler. The concept is inspired by the strangler fig tree, which grows around an existing tree, gradually replacing it. Applied to software, this pattern involves wrapping existing legacy components with modern code and gradually replacing them piece by piece. It allows teams to modernize systems incrementally while maintaining a functioning product. In Blackboard’s case, the engineering team began by introducing RESTful APIs that abstracted access to the legacy system. These APIs provided a clean interface between the old and new code, enabling developers to build new features in microservices that communicated with both legacy and modern components. This method allowed Blackboard to improve the system piece by piece without ever taking it offline or losing critical functionality.

The adoption of the strangler pattern was not just a technical shift but a cultural one as well. Blackboard embraced DevOps principles, emphasizing collaboration between development and operations teams, automated testing, continuous integration, and continuous delivery. Development teams were given full ownership of the services they built, creating a stronger sense of accountability and autonomy. This cultural shift enabled faster decision-making, quicker delivery of new features, and a more resilient development process overall. Blackboard also invested in modern tooling and infrastructure, such as CI/CD pipelines and robust monitoring systems. These tools supported the slow, deliberate modernization of the platform and helped prevent regressions during the transition. Through careful planning and execution, the company managed to improve reliability, scalability, and agility all without risking the stability of its product.

The case study of Blackboard Learn is a powerful example of how large organizations can modernize legacy systems without halting operations or incurring massive risk. By applying the strangler pattern and embracing DevOps culture, Blackboard achieved a gradual but sustainable transformation. This approach demonstrates that modernization is not about starting over, it’s about working smarter, using proven techniques to evolve what you already have.

Several important lessons emerged from Blackboard’s successful use of the strangler pattern:

* Avoid the Big Rewrite: Rebuilding a complex system from scratch is risky, expensive, and usually unsuccessful. The strangler approach allowed Blackboard to reduce risk by delivering value incrementally.
* Modular Architecture Is Critical: Decoupling the monolith into smaller services made the system more maintainable and allowed teams to innovate independently.
* Continuous Delivery Enables Success: Without modern DevOps practices like automation and CI/CD, a gradual transition would have been far more difficult to manage.
* Team Empowerment Drives Results: Giving teams ownership over their services improved morale and accountability, leading to better software quality and faster delivery.

Reference:

Kim, G., Humble, J., Debois, P., & Willis, J. (2021). The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations (2nd ed.). IT Revolution Press.